UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/663,563	09/15/2000	Thomas E. Saulpaugh	5181-72000	3700

7590 06/23/2005

Robert C Kowert
conley Rose & Tayon PC
PO Box 398
Austin, TX 78767-0398

EXAMINER	
SHRADER, LAWRENCE J	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 06/23/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/663,563	SAULPAUGH ET AL.	
	Examiner	Art Unit	
	Lawrence Shrader	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 February 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-90* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-90* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *2/22/2005*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This office action is in response to the Applicant's amendment filed on 2/22/2005.

2. The Applicant's arguments presented in the amendment have been fully considered, but are moot in view of the new grounds of rejection.

Information Disclosure Statement

3. The Information Disclosure Statement filed on 2/22/2005 is acknowledged, and it has been considered.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 9 – 11; 25, 27 – 28, 30 – 33; 34 – 39; 40, 41, 44, 46 – 49; 50, 51, 58 – 61; 62 – 66, 69, 70; 71, 75 – 77; 84 – 90 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tuatini, U.S. Patent Application US2002/0035645.

Tuatini discloses a method for representing JAVA objects in XML

In regard to claim 1:

"a process executing within a virtual machine providing a first computer programming language object to a compilation process of the virtual machine, wherein the first object is an instance of a class in the computer programming language;"

"the compilation process of the virtual machine converting the first object into a data representation language representation of the first object;"

"wherein the data representation language representation of the first object is configured for use in generating a copy of the first object."

See Tuatini for providing a JAVA object to a compiler (para. [0079]). XML (data representation language) is convertible to and from JAVA objects. The cited reference in the Tautini invention clearly discloses converting XML to JAVA objects and JAVA objects to an XML representation. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to apply this well-known conversion technique to generate a copy of a JAVA object by converting it to a data representation language like XML because XML is known to be a form of self-describing data having utility as a data exchange medium between dissimilar systems where a Java development process can be simplified by using XML to deliver data and to describe the business objects that are responsible for building the data.

In regard to claims 9 - 11, incorporating the rejection of claim 1:

"...wherein said data representation language is eXtensible Markup Language (XML)."

"...wherein said computer programming language is the Java programming language."

"...wherein the virtual machine is a Java Virtual Machine (JVM)."

Tuatini discloses XML, JAVA programming language and JAVA environment running on a virtual machine (Figures 2 and 3; para. [0079]).

In regard to claim 25:

Tuatini discloses a method for passing computer programming language objects between processes in a distributed computing environment, comprising:

> *"a first virtual machine receiving from a first process a computer programming language object, wherein the object is an instance of a class in the computer programming language;*
>
> *the first virtual machine generating a representation of the object in a data representation language subsequent to said receiving;*
>
> *generating a message in the data representation language, wherein the message includes the data representation language representation of the object;*
>
> *sending the message to a second process; and*
>
> *the second process generating a copy of the computer programming language object from the data representation language representation of the object included in the message."*

Claim 25 is rejected for the same reasons put forth in the rejection of claim 1. The only difference between claim 1 and claim 25 is that the object is transmitted over the network (sending the message) and then converted back to XML, which finds support in Tuatini at Figures 3 and 4 where the process is conducted over a localized distributed environment. See Tuatini for providing a JAVA object to a compiler (para. [0079]). XML (data representation language) is convertible to and from JAVA objects. The cited reference in the Tautini invention clearly discloses converting XML to JAVA objects and JAVA objects to an XML representation.

Therefore, it would have been obvious to one skilled in the art at the time the invention was made to apply this well-known conversion technique to generate a copy of a JAVA object by converting it to a data representation language like XML because XML is known to be a form of self-describing data having utility as a data exchange medium between dissimilar systems where a Java development process can be simplified by using XML to deliver data and to describe the business objects that are responsible for building the data.

In regard to claim 27, a method incorporating the rejection of claim 25:

> *"...wherein the object comprises one or more instance variables, and wherein said generating a representation of the object in a data representation language comprises:*
>
> *for each of the one or more instance variables in the object, generating an element in the data representation language representation of the first object, wherein the element for each of the one or more instance variables includes an identifier of the instance variable and a value of the instance variable."*

Tuatini discloses a process executing on a virtual machine producing a JAVA object and converting it to XML (para. [79])

In regard to claim 28, a method incorporating the rejection of claim 25, it is rejected for the same corresponding reasons put forth in the rejection of claim 1.

In regard to claim 30, incorporating the rejection of claim 28:

> *"...wherein the data representation language representation of the object comprises one or more elements each representing an instance variable of the object, and wherein said generating the copy of the object from the data representation language representation of the object comprises:*
>
> *instantiating the copy of the object as an instance of the class; and*
>
> *for each of the one or more elements in the data representation language representation of the object, initializing a corresponding instance variable in the copy of the object in accordance with the element."*

Tuatini discloses a process executing on a virtual machine instantiating JAVA objects having class variables (instance variable) that encapsulate the data structures (elements) corresponding to the information in the data representation language, i.e., the XML (see Figure 18).

In regard to claim 31, a method incorporating the rejection of claim 25, it is rejected for the same corresponding reasons put forth in the rejection of claim 9.

In regard to claim 32, a method incorporating the rejection of claim 25, it is rejected for the same corresponding reasons put forth in the rejection of claim 10.

In regard to claim 33, a method incorporating the rejection of claim 25, it is rejected for the same corresponding reasons put forth in the rejection of claim 11.

In regard to claim 34, it is rejected for the same corresponding reasons put forth in the rejection of claim 25. It contains the same inter process exchange of information.

In regard to claim 35, a method incorporating the rejection of claim 34, it is rejected for the same corresponding reasons put forth in the rejection of claim 27.

In regard to claim 36, incorporating the rejection of claim 35:

> *"...wherein said generating the object from the 30 information representing the object comprises:*
>
> *instantiating the object as an instance of the class; and*
>
> *for each of the one or more instance variables, initializing a corresponding instance variable in the object in accordance with the information representing the instance variable."*

Tuatini discloses a process executing on a virtual machine instantiating JAVA objects having class variables (instance variable) that encapsulate the data structures (elements) corresponding to the information in the data representation language, i.e., the XML (see Figure 18).

In regard to claim 37 (a device) incorporating the rejection of claim 34, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 9.

In regard to claim 38 (a device) incorporating the rejection of claim 34, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 10.

In regard to claim 39 (a device) incorporating the rejection of claim 34, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 11.

In regard to claim 40 (a device), it is rejected for the same corresponding reasons put forth in the rejection of the method claim 1.

In regard to claim 41 (a device) incorporating the rejection of claim 40, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 2.

In regard to claim 44 (a device) incorporating the rejection of claim 43, it is rejected for the same corresponding reasons put forth in the rejection of the method claims 6, and 7.

In regard to claim 46 (a device) incorporating the rejection of claim 40, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 9.

In regard to claim 47 (a device) incorporating the rejection of claim 40, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 10.

In regard to claim 48 (a device) incorporating the rejection of claim 40, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 11.

In regard to claim 49 (a device) incorporating the rejection of claim 48, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 11.

In regard to claim 50 (a device), it is rejected for the same corresponding reasons put forth in the rejection of the method claim 1. The virtual machine processing instruction is found at column 5, lines 18 – 31.

In regard to claim 51 (a device) incorporating the rejection of claim 50, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 1. The virtual machine processing instructions is found at column 5, lines 18 – 31.

In regard to claim 58, (a device) incorporating the rejection of claim 50, it is rejected for the same corresponding reasons put forth in the rejection of claim 9.

In regard to claim 59, (a device) incorporating the rejection of claim 50, it is rejected for the same corresponding reasons put forth in the rejection of claim 10.

In regard to claim 60, (a device) incorporating the rejection of claim 50, it is rejected for the same corresponding reasons put forth in the rejection of claim 11.

In regard to claim 61, (a device) incorporating the rejection of claim, it is rejected for the same corresponding reasons put forth in the rejection of claim 11.

In regard to claim 62 (a distributed computing system), it is rejected for the same corresponding reasons put forth in the rejection of claim 25.

In regard to claim 63 (a distributed computing system) incorporating the rejection of claim 62, it is rejected for the same corresponding reasons put forth in the rejection of claim 25.

In regard to claim 64 (a distributed computing system) incorporating the rejection of claim 63, it is rejected for the same corresponding reasons put forth in the rejection of claim 25.

In regard to claim 65 (a distributed computing system) incorporating the rejection of claim 63, it is rejected for the same corresponding reasons put forth in the rejection of claim 25. Tuatini inherently discloses a JAVA virtual machine as a container environment in Figures 2 and 3, which processes byte code into program instructions.

In regard to claim 66 (a distributed computing system) incorporating the rejection of claim 63, it is rejected for the same corresponding reasons put forth in the rejection of claim 11.

In regard to claim 69 (a distributed computing system) incorporating the rejection of claim 62, it is rejected for the same corresponding reasons put forth in the rejection of claim 9.

In regard to claim 70 (a distributed computing system) incorporating the rejection of claim 62, it is rejected for the same corresponding reasons put forth in the rejection of claim 10.

In regard to claim 71 (a carrier medium), it is rejected for the same corresponding reasons put forth in the rejection of the method claim 1.

In regard to claim 75 (a carrier medium) incorporating the rejection of claim 71, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 9.

In regard to claim 76 (a carrier medium) incorporating the rejection of claim 71, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 10.

In regard to claim 77 (a carrier medium) incorporating the rejection of claim 71, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 11.

In regard to claim 84 (a carrier medium), it is rejected for the same corresponding reasons put forth in the rejection of the method claim 25.

In regard to claim 85 (a carrier medium) incorporating the rejection of claim 84, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 27.

In regard to claim 86 (a carrier medium) incorporating the rejection of claim 84, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 28.

In regard to claim 87 (a carrier medium) incorporating the rejection of claim 86, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 30.

In regard to claim 88 (a carrier medium) incorporating the rejection of claim 84, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 31.

In regard to claim 89 (a carrier medium) incorporating the rejection of claim 84, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 32.

In regard to claim 90 (a carrier medium) incorporating the rejection of claim 84, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 33.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 2; 26, 29; 52; and 67 – 68 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tuatini, U.S. Patent Application US2002/0035645 in view of Mitchell, U.S. Patent 6,628,304 (hereinafter referred to as Mitchell).

In regard to claim 2, incorporating the rejection of claim 1:

> *"...wherein the first object references one or more computer programming language objects, and wherein the compilation process of the virtual machine converting the first object into a data representation language representation of the first object comprises the compilation process converting the one or more objects into data representation language representations of the one or more objects."*

Tuatini discloses that the XML is convertible to and from JAVA objects (para. [0079]), but does not explicitly disclose that the first object may reference more than one object. However, Mitchell discloses one object representing a hierarchy of multiple other objects (column5, lines 38 – 55). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the process of translation of XML to JAVA in the Tuatini invention with the teaching of Mitchell because the first object of Tuatini would then represent one or more objects of a hierarchical relationship as taught by Mitchell (column 5, lines 38 – 43; column 20, lines 43 – 47) and corresponds to the hierarchical nature of XML text which the object would represent.

In regard to claim 26, a method incorporating the rejection of claim 25, it is rejected for the same corresponding reasons put forth in the rejection of claim 2.

In regard to claim 29, a method incorporating the rejection of claim 28, it is rejected for the same corresponding reasons put forth in the rejection of claim 2.

In regard to claim 52, (a device) incorporating the rejection of claim 50, it is rejected for the same corresponding reasons put forth in the rejection of claim 2.

In regard to claim 67 (a distributed computing system) incorporating the rejection of claim 62, it is rejected for the same corresponding reasons put forth in the rejection of claim 29.

In regard to claim 68 (a distributed computing system) incorporating the rejection of claim 62, it is rejected for the same corresponding reasons put forth in the rejection of claim 30.

8. Claims 3 – 7; 42 – 44; 53 – 55; and 72 – 74 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tuatini, U.S. Patent Application US2002/0035645 in view of Humpleman et al. U.S. Patent 6,546,419 (hereinafter referred to as Humpleman).

In regard to claim 3, incorporating the rejection of claim 1:

> *"...wherein the compilation process of the virtual machine converting the first object into a data representation language representation of the first object comprises:*
>
> *processing the first object into an intermediary table representation of the first object; and*
>
> *processing the intermediary table representation of the first object into the data representation language representation of the first object."*

Tuatini discloses a process executing on a virtual machine producing a JAVA object and converting it to XML ([para. 0079]), but does not disclose compiling the object into an

intermediary table, which in turn is processed to produce the data representation language. However, Humpleman discloses an intermediary table created at compile time for the purpose of converting XML (data representation language) messages to C language and vice versa (column 12, lines 55 – 67; e.g., Figure 15). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the teachings of Tuatini, producing a JAVA object converted to XML, and Humpleman wherein the conversion of the of the object to XML is done through a table because the combination provides run-time translation of XML into native language as taught by Humpleman at column 12, lines 63 – 65.

In regard to claim 4, incorporating the rejection of claim 3:

> *"... wherein the first object comprises one or more instance variables, and wherein said processing the first object into an intermediary table representation comprises:*
>
> *for each of the one or more instance variables in the first object, generating an entry in the intermediary table representation of the first object, wherein the entry for each of the one or more instance variables includes an identifier of the instance variable and a value of the instance variable."*

Tuatini discloses a process executing on a virtual machine producing a JAVA object convertible to an XML element (para. [0079]), but does not disclose compiling the object into an intermediary table, which in turn is processed to produce the data representation language. However, Humpleman discloses an intermediary table created at compile time for the purpose of converting XML (data representation language) messages to C language and vice versa, with class variables (instance variable) that are inherently identified with a value used to generate the intermediary table (column 13, lines 9 – 17; e.g., Figure 15). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the teachings of Tuatini, producing a JAVA object converted to XML, and Humpleman wherein the conversion

of the of the object to XML is done through a table with the instance variable identified with a value in order to provide run-time translation of XML into native language as taught by Humpleman at column 12, lines 63 – 65.

In regard to claim 5, incorporating the rejection of claim 4:

> *"...wherein the first object comprises a plurality of instance variables with the same identifier, and wherein the entry for each of the plurality of instance variables with the same identifier further includes an enumeration value that uniquely identifies the instance variable in the plurality of instance variables with the same identifier."*

Tuatini discloses a process executing on a virtual machine producing a JAVA object convertible to an XML element (para. [0079]), but does not disclose compiling the object into an intermediary table, which in turn is processed to produce the data representation language. However, Humpleman discloses an intermediary table created at compile time for the purpose of converting XML (data representation language) messages to C language and vice versa, with class variables (instance variable) that are inherently identified with an instance value used to generate the intermediary table (column 13, lines 9 – 17; e.g., Figure 15). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the teachings of Tuatini, producing a JAVA object converted to XML, and Humpleman wherein the conversion of the of the object to XML is done through a table with the instance variable identified with a value in order to provide run-time translation of XML into native language as taught by Humpleman at column 12, lines 63 – 65.

In regard to claim 6, incorporating the rejection of claim 4:

> *"...wherein said processing the intermediary table representation of the first object into the data representation language representation of the first object comprises:*
>
> *for each of one or more entries in the intermediary table representation of the first*

object, generating a corresponding element in the data representation language representation of the first object, wherein the element in the data representation language representation of the first object includes an identifier of the instance variable and a value of the instance variable."

Tuatini discloses a process executing on a virtual machine producing a JAVA object convertible to an XML element (para. [0079]), but does not disclose compiling the object into an intermediary table, which in turn is processed to produce the data representation language. However, Humpleman discloses an intermediary table created at compile time for the purpose of converting XML (data representation language) messages to C language and vice versa, with class variables (instance variable) that is inherently identified with a value used to generate the intermediary table (column 13, lines 9 – 17; e.g., Figure 15). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the teachings of Tuatini, producing a JAVA object converted to XML, and Humpleman wherein the conversion of the of the object to XML is done through a table with the instance variable identified with a value in order to provide run-time translation of XML into native language as taught by Humpleman at column 12, lines 63 – 65.

In regard to claim 7, incorporating the rejection of claim 6:

"... wherein the one or more elements in the data representation language representation of the first object are configured for use in initializing one or more corresponding instance variables in the copy of the first object."

XML can be converted back to JAVA objects (para. [0079]).

In regard to claim 42 (a device) incorporating the rejection of claim 40, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 3.

In regard to claim 43 (a device) incorporating the rejection of claim 42, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 4.

In regard to claim 44 (a device) incorporating the rejection of claim 43, it is rejected for the same corresponding reasons put forth in the rejection of the method claims 6, and 7.

In regard to claim 53, (a device) incorporating the rejection of claim 50, it is rejected for the same corresponding reasons put forth in the rejection of claim 3.

In regard to claim 54, (a device) incorporating the rejection of claim 50, it is rejected for the same corresponding reasons put forth in the rejection of claim 4.

In regard to claim 55, (a device) incorporating the rejection of claim 54, it is rejected for the same corresponding reasons put forth in the rejection of claim 4.

In regard to claim 72 (a carrier medium) incorporating the rejection of claim 71, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 3.

In regard to claim 73 (a carrier medium) incorporating the rejection of claim 72, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 4.

In regard to claim 74 (a carrier medium) incorporating the rejection of claim 73, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 6.

9. Claims 8, 45, and 57 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tuatini, U.S. Patent Application US2002/0035645 in view of Daly et al., U.S Patent 5,748,896 (hereinafter referred to as Daly).

In regard to claim 8, incorporating the rejection of claim 1:

> *"...further comprising providing an application programming interface (API) for the compilation process, wherein the API comprises interfaces to one or more methods of the compilation process configured for use by processes executing within the virtual machine to convert computer programming language objects into data representation language representations of the objects."*

Tuatini discloses a process executing on a virtual machine producing a JAVA object, but does not disclose compiling with an API. However, Daly discloses compiling through the use of an API (column 16, lines 64 – 65). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the teachings of Tuatini with compiling using an API as taught by Daly because it would provide executable interface code that can be invoked without recompiling or linking and providing transportability across platforms.

In regard to claim 45 (a device) incorporating the rejection of claim 44, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 8.

In regard to claim 57, (a device) incorporating the rejection of claim 51, it is rejected for the same corresponding reasons put forth in the rejection of claim 8.

10. Claims 12; 20, 22 – 24; 78, and 81 – 83 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tuatini, U.S. Patent Application US2002/0035645 in view of Ansari et al., U.S Patent 5,881,290 (hereinafter referred to as Ansari).

In regard to claim 12:

Tuatini discloses a method for generating JAVA objects from XML (para. [0079]).

"a virtual machine receiving a data representation language representation of a first computer programming language object from a first process;"

See Tuatini Figure 33 for reception of XML (data representation language representation).

"a decompilation process of the virtual machine generating the first object from the data representation language representation of the first object, wherein the first object is an instance of a class in the computer programming language;"

"the decompilation process of the virtual machine providing the first object to a second process executing within the virtual machine."

Tuatini discloses reception of XML (a data representation language) and translating to JAVA objects, but does not disclose the translation as a decompilation process, although this process could be implied. However, Ansari explicitly discloses a method of compiling and decompiling so that the original program information can undergo some processing (column 3, lines 44 – 47; column 9, lines 34 – 38). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the process of translation of XML to JAVA in the Tuatini invention with the decompiling function of Ansari because the combination provides a means for the Tuatini invention to extract the object information so that it can be

processed as taught Ansari at column 3, lines 46 – 47, where one can interpret "editing or the like" to comprise the use of another process.

In regard to claim 20, incorporating the rejection of claim 12:

> *"...wherein the data representation language representation of the first object comprises an identifier of the class of the first object, and wherein the decompilation process generating the first object from the data representation language representation of the first object comprises instantiating the first object as an instance of the class associated with the class identifier."*

Tuatini discloses reception of XML (a data representation language) and translating to compiled JAVA objects (para. [0079]), but does not disclose the reverse translation as a decompilation process, although this process could be implied. However, Ansari explicitly discloses a method of compiling and decompiling so that the original program information can undergo some processing (column 3, lines 44 – 47; column 9, lines 34 – 38). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the process of translation of XML to JAVA and vice versa in the Tuatini invention with the decompiling function of Ansari because the combination provides a means for the Tuatini invention to extract the object information so that it can be processed as taught Ansari at column 3, lines 46 – 47, where one can interpret "editing or the like" to comprise the use of another process.

In regard to claims 22 - 24, incorporating the rejection of claim 12:

> *"...wherein said data representation language is eXtensible Markup Language (XML)."*
>
> *"...wherein said computer programming language is the Java programming language."*
>
> *"...wherein the virtual machine is a Java Virtual Machine (JVM)."*

Tuatini discloses XML and JAVA programming language and a JAVA virtual machine architecture is inherently present (para. [0079]).

In regard to claim 78 (a carrier medium), it is rejected for the same corresponding reasons put forth in the rejection of the method claim 12.

In regard to claim 81 (a carrier medium) incorporating the rejection of claim 78, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 22.

In regard to claim 82 (a carrier medium) incorporating the rejection of claim 78, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 23.

In regard to claim 83 (a carrier medium) incorporating the rejection of claim 78, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 24.

11. Claims 13 and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tuatini, U.S. Patent Application US2002/0035645 in view of Ansari et al., U.S Patent 5,881,290, and further in view of Mitchell, U.S. Patent 6,628,304.

In regard to claim 13, incorporating the rejection of claim 12:

> *"...wherein the first object references one or more computer programming language objects, wherein the representation of the first object includes representations of the one or more referenced objects."*

Tuatini discloses XML (data representation language) convertible to and from JAVA objects (para. [0079]), but neither Tuatini nor Ansari discloses the first object representing one or more referenced objects. However, Mitchell discloses one object representing a hierarchy of multiple other objects (column5, lines 38 – 55). Therefore, it would have been obvious to one

skilled in the art at the time the invention was made to combine the process of translation of XML to JAVA in the Tuatini invention with the decompiling function of Ansari because the combination provides a means for the Tuatini invention to extract the object information so that it can be processed as taught Ansari at column 3, lines 46 – 47, and further combined with the teaching of Mitchell because the first object may then represent one or more objects of a hierarchical relationship as taught by Mitchell (column 5, lines 38 – 43; column 20, lines 43 – 47).

In regard to claim 14, incorporating the rejection of claim 13:

> *"...wherein the decompilation process of the virtual machine generating the first object from the representation of the first object comprises the decompilation process generating the one or more referenced objects from the representations of the one or more referenced objects included in the representation of the first object."*

Tuatini discloses XML (data representation language) convertible to and from JAVA objects and Ansari explicitly discloses a method of compiling and decompiling so that the original program information can undergo some processing, but neither Tuatini nor Ansari discloses the first object representing one or more referenced objects. However, Mitchell discloses one object representing a hierarchy of multiple other objects (column5, lines 38 – 55). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the process of translation of XML to JAVA in the Tuatini invention with the decompiling function of Ansari because the combination provides a means for the Tuatini invention to extract the object information so that it can be processed as taught Ansari at column 3, lines 46 – 47, and further combined with the teaching of Mitchell wherein the objects may represent one or more objects of a hierarchical relationship as taught by Mitchell (column 5, lines 38 – 43; column 20, lines 43 – 47).

12. Claims 15 – 19; 56; 79 and 80 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tuatini, U.S. Patent Application US2002/0035645 in view of Ansari et al., U.S Patent 5,881,290, and further in view of Humpleman et al. U.S. Patent 6,546,419 (hereinafter referred to as Humpleman).

In regard to claim 15, incorporating the rejection of claim 12:

> *"...wherein the decompilation process generating the first object from the data representation language representation of the first object comprises:*
>
> *processing the data representation language representation of the first object into an intermediary table representation of the first object;*
>
> *generating the first object from the intermediary table representation of the first object."*

Tuatini discloses reception of XML (a data representation language), but does not disclose the translation as decompilation, although this process could be implied. However, Ansri explicitly discloses a method of compiling and decompiling so that the original program information can undergo some processing. Neither Tuatini nor Ansari discloses a decompiling process generating a first object into an intermediary table, which in turn is processed to generate the first object. However, Humpleman discloses an intermediary table created at compile time for the purpose of converting XML (data representation language) messages to C language and vice versa (column 13, lines 9 – 17; e.g., Figure 15, Figure 16 – objects are transferred). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the process of translation of XML to JAVA in the Tuatini invention with the

decompiling function of Ansari because the combination provides a means for the Tuatini invention to extract the object information so that it can be processed, and further modified by Humpleman wherein the conversion of the of the object is done through a table in order to provide run-time translation of XML into native language as taught by Humpleman at column 12, lines 63 – 65 providing a means to recover and process the object of concern.

In regard to claim 16, incorporating the rejection of claim 15:

> *"...wherein the data representation language representation of the first object comprises one or more elements each representing an instance variable of the first object, wherein each element in the data representation language representation comprises an identifier for the instance variable represented by the element and a value for the instance variable represented by the element."*

Tuatini discloses a process executing on a virtual machine producing JAVA objects having class variables (instance variable) that encapsulate the data structures (elements) corresponding to the information in the data representation language, i.e., the XML (para. [0079]).

In regard to claim 17, incorporating the rejection of claim 16:

> *"...wherein said processing the data representation language representation of the first object into an intermediary table representation of the first object comprises generating an entry in the intermediary table representation of the first object for each of the one or more elements in the data representation language 5 representation of the first object."*

Tuatini discloses a process executing on a virtual machine producing JAVA objects having class variables (instance variable) that encapsulate the data structures (elements) corresponding to the information in the data representation language, i.e., the XML, but does not disclose compiling the object into an intermediary table, which in turn is processed to produce the data representation language. However, Humpleman discloses an intermediary table created

at compile time for the purpose of converting XML (data representation language) messages to C language and vice versa, with class variables (instance variable) that are inherently identified with a value used to generate the intermediary table (column 13, lines 9 – 17; e.g., Figure 15). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the teachings of Tuatini, producing a JAVA object converted to XML, and Humpleman wherein the conversion of the of the object to XML is done through a table with the instance variable identified with a value in order to provide run-time translation of XML into native language as taught by Humpleman at column 12, lines 63 – 65.

In regard to claim 18, incorporating the rejection of claim 17:

> *"...wherein said generating the first object from the intermediary table representation of the first object comprises:*
>
> *instantiating the first object as an instance of the class; and*
>
> *for each of the one or more entries in the intermediary table representation of the first object, initializing a corresponding instance variable in the first object in accordance with the entry."*

Tuatini discloses a process executing on a virtual machine instantiating JAVA objects having class variables (instance variable) that encapsulate the data structures (elements) corresponding to the information in the data representation language, i.e., the XML (para. [0079]).

In regard to claim 19, incorporating the rejection of claim 17:

> *"...wherein said processing the intermediary table representation of the first object into the first object comprises:*
>
> *instantiating the first object as an instance of the class; and*

for each of the one or more entries in the intermediary table representation of the first object, invoking a method corresponding to the identifier of the instance variable from the entry to initialize a corresponding instance variable in the first object to the value of the instance variable from the entry."

Tuatini discloses a process executing on a virtual machine producing JAVA objects having class variables (instance variable) that encapsulate the data structures (elements) corresponding to the information in the data representation language, i.e., the XML, but does not disclose compiling the object into an intermediary table, which in turn is processed to produce the data representation language. However, Humpleman discloses an intermediary table created at compile time for the purpose of converting XML (data representation language) messages to C language, and vice versa, with class variables (instance variable) that are inherently identified with a value used to generate the intermediary table (column 13, lines 9 – 17; e.g., Figure 15). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the teachings of Tuatini, producing a JAVA object converted to XML having class variables (instance variable) that encapsulate the data structures (elements) corresponding to the information in the data representation language, i.e., the XML (see column 22, lines 46 – 65), and Humpleman wherein the conversion of the of the object to XML is done through a table with the instance variable identified with a value in order to provide run-time translation of XML into native language as taught by Humpleman at column 12, lines 63 – 65.

In regard to claim 56, (a device) incorporating the rejection of claim 55, it is rejected for the same corresponding reasons put forth in the rejection of claim 19.

In regard to claim 79 (a carrier medium) incorporating the rejection of claim 78, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 15.

In regard to claim 80 (a carrier medium) incorporating the rejection of claim 79, it is rejected for the same corresponding reasons put forth in the rejection of the method claim 18.

13. Claim 21 is rejected under 35 U.S.C. 103(a) as being unpatentable over Tuatini, U.S. Patent Application US2002/0035645 in view of Ansari et al., U.S Patent 5,881,290, and further in view of Daly et al., U.S Patent 5,748,896.

In regard to claim 21, incorporating the rejection of claim 12:

> *"...further comprising providing an application programming interface (API) for the decompilation process, wherein the API comprises interfaces to one or more methods of the decompilation process configured for use by processes executing within the virtual machine to generate computer programming language objects from data representation language representations of the objects."*

Neither Tuatini nor Ansari discloses an API comprising interfaces to one or more methods of the decompilation process on a virtual machine producing. However, Daly discloses compiling through the use of an API (column 16, lines 64 – 65). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the process of translation of XML to JAVA and vice versa in the Tuatini invention with the decompiling function of Ansari because the combination provides a means for the Tuatini invention to generate the object information, and further combined with the teaching of Daly, because the combination provides executable interface code that can be invoked without recompiling or linking and providing transportability across platforms.

Response to Arguments

14. Applicant's arguments filed on 2/22/2005 have been fully considered but they are not persuasive:

The Applicant has argued:

> Tuatini's system then "deserializes the message and returns the JAVA object representing the message" and "performs any processing necessary to convert the message from the client format to the application format" (Tuatini, page 10, paragraph 0083-0084). Thus, any XML response message serialized from one of Tuatini's objects is not a data representation language representation of a computer programming language object, but instead is an XML response message in a client format different from the application format used by the object. Tuatini's system converts messages between formats to enable communication across platforms and between new and legacy applications.
>
> Furthermore, in Tuatini an XML response message in a client format converted from a JAVA object in an application format is not a data representation language representation of the JAVA object. Nor is it configured for use in generating a copy of the JAVA object. Tuatini teaches that the data in the XML message is in a different format from the data in the JAVA object and thus is not configured to use is generating a copy of the object. Tuatini does not teach a data representation language representation of a first computer programming language object wherein the data representation language representation is configured for use in generating a copy of the first object.

Examiner's response:

There is ample documentation in the literature for converting objects to XML and back again, e.g., see the pertinent references supplied in the conclusion. In other words, the process of converting of converting a JAVA object to XML and then back to a JAVA object produces a copy of the first object, and is well known in the art. Therefore, a broad and reasonable interpretation of the cited passage from Tuatini, paragraph [0079] reads on the claims as written.

Conclusion

15. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure:

> U.S. Patent 6,634,008 to Dole, regarding conversion of a methodology into XML and converting the XML into multiple files.
>
> "An XML Document to JavaScript Object Converter," Alexander Hildyard, Web Techniques, Volume 4, Issue 1 (January 1999).
>
> "Integrating XML and Object-based Programming for Distributed Collaboration," Roussev, et al., Proceedings. IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000. (WET ICE 2000), 14 – 16 June 2000, pp. 254 – 259.

16. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lawrence Shrader whose telephone number is (571) 272-3734. The examiner can normally be reached on M-F 08:00-16:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Lawrence Shrader
Examiner
Art Unit 2193

10 June 2005

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100